Assistive Mobile System Design for Tracking Small Industrial Assets

Harikrishna G. N. Rai, K. Sai Deepak
Infosys Labs, Infosys Limited
Bangalore, India
(harikrishna_rai, krishnamurtysai_d}@infosys.com

Dinesh Godavarthi, Shahnaz Syed, P. Radha Krishna
Infosys Limited
Hyderabad, India
{dinesh_godavarthi, shahanaz_syed, radhakrishna_p}@infosys.com

***Abstract*—Advancement of machine vision technologies in mobile space is opening new avenues for industrial automation businesses. One such area is tracking small industrial assets in manufacturing plants where implementing conventional technologies is not viable. Managing the excess components on assembly floor after assembling the products is a challenging task, especially locating their original place in the store keeping units. In this paper, we propose a system to assist store keepers for efficiently tracking and storing small assets. The system is developed using a computer vision based mobile application for asset identification. We use content based image retrieval technology to retrieve asset images based on their visual similarity from a database. A design methodology and architecture for the mobile application is also described in this paper. Experimental results on an industrial asset image dataset are presented to demonstrate our proposed approach.**

Keywords-mobile application, computer vision, asset tracking, content based image retrieval, feature extraction

I. Introduction

In every industrial sector, business stakeholders are continuously looking for technological evolutions that can help in improving their business processes. Early adopters of technology are going to win in the competitive business world; hence there is a continuous push for adopting advanced technologies. Business sectors such as manufacturing, industrial automation and automotive are among the forerunners in adopting machine vision technologies [1]. Due to the advancements in mobile technology, business automation has become simpler, efficient and faster. Mobile enabled industrial business processes helps businesses in optimizing their complex workflows thereby increasing profitability.

Industrial asset management and tracking is very crucial to business and there are several opportunities for mobile based asset management and tracking processes. Efficient asset management helps industries to reduce their asset loss, proper utilization of costlier assets, maintain inventory level required etc. Another important aspect of efficient asset management is the ability to track the location of an industrial asset which is presently done manually and it is a time consuming and error prone process. Ability of asset management systems to locate the asset position in real-time as and when required is key to business success especially in product lines where visibility and quick access to assets are most important to ensure quick production time. Misplacement of necessary components and lack of required number of components in assembly process can cause large revenue loss due to delay in production stage [2]. Subsequent delays in each intermediate stage of productions gets added up when production process is completed and this will impact the final delivery of product to the customers. To address above challenges, there are a few existing technologies being adopted for tracking industrial assets. We explore the state-of-the-art industrial asset tracking technology and discuss the challenges in implementing them in the following sections.

A. *Present technologies for industrial asset tracking*

Bar codes and *Radio Frequency Identification (RFID)* are used for effective inventory management of spare parts and components. The popular bar code technique uses unique bar code patterns for each component where the codes are mapped to component details such as make, model number and location in a computer system. Using a bar code

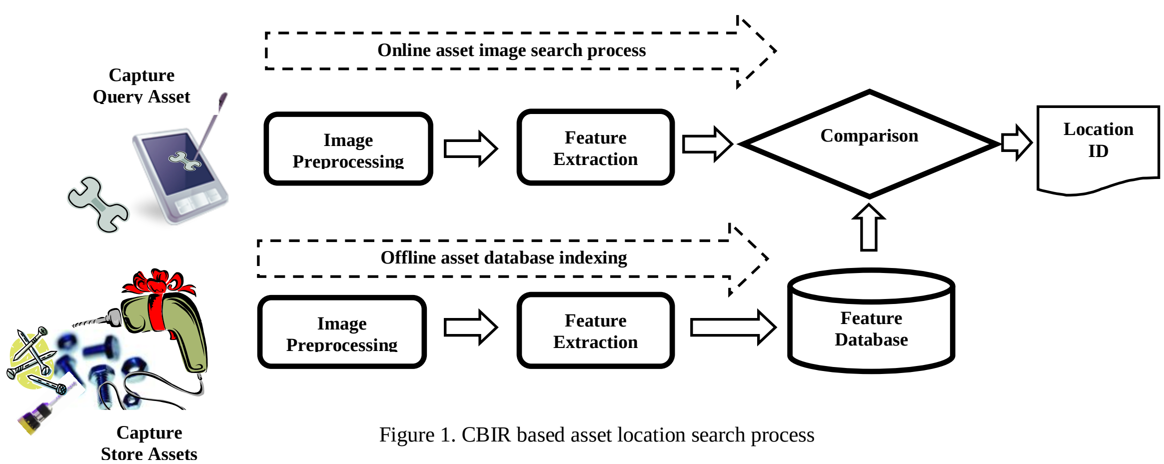

Figure 1. CBIR based asset location search process

reader, store keeper captures the barcode and the encoded details are retrieved from the system. RFID is another alternate technology widely used for component and asset tracking in industries. RFID based solutions have several advantages such as cheap, fast, more secure and reliable over manual bar code based component location tracking. Unlike barcode, RFID based location tracking systems do not need line-of-sight for detecting components. Hence, even if the components are in larger heaps without direct visibility to human eyes, RFID systems can still detect components using embedded sensors. In this system RFID tags are fitted with the component to be located. RFID tags serve as radio frequency transmitters that comprise a microchip, an antenna and a tiny battery. RFID receiver detects the RF signals emitted by each tag and interprets the encoded identification details. Several solutions based on RFID technology (ex., *RollCall* [3] and *Smart Tracker* [4]) were proposed in the literature. However, RFID based systems have some known limitations (see section I.C) that makes them inappropriate in the context of small asset tracking. In the following section, we discuss the challenges in tracking small industrial assets on assembly floor in detail and reveal the limitations of existing tracking techniques in this context.

B. Challenges in tracking small industrial assets

Managing the excess components remaining after assembling a product on an assembly floor has been a challenging task in manufacturing sector. In an assembly line, machine tools, spare parts, mechanical components such as bearings, nuts and bolts are used in large numbers and they are availed in large quantity from store rooms than their actual requirements. So, excess components are left out on assembly floor after completion of production, and store keepers have to pick them from assembly floor and place them back to their original place in store keeping units (SKUs) maintained at store rooms. Identifying the original location of excess components exactly in SKUs is a tedious task, which results in misplacement of components and eventually resulting in ineffective inventory management. Since inventory management system shows that there are required numbers of components available in SKU but, when needed, components may not exist in the actual location. Usually, assembly line managers will plan assembly process by referring the inventory status. However, due to misplaced components, there is a delay in manufacturing time in locating the components, which may result in great business loss to manufacturing organizations.

C. Limitations of RFID and Barcode technology

One severe limitation of barcode and RFID based technologies for small components is that they may not have enough surface area to attach either bar code or RFID tag. Also basic components such as screws and bolts are used in large quantity and so it is a tedious task to manually process each component while archiving in store room. One solution to overcome it is by collecting all smaller components in one container and attaching the bar code to container, but this also time consuming and may not help in identifying the components once brought outside the container.

Unlike RFID and bar code, machine vision systems usually does not require any identifiers to be attached on the assets. In this work, we explore the possibility of devising a highly mobile, practical and usable vision based application for tracking and consolidation of small industrial assets. Industrial automation sector is always ahead in adopting machine vision technology [1] and already they are using such technology for industry process automation and quality inspection. Hence machine vision based solutions are easy to deploy in the existing environment.

In our earlier work [5], we introduced the concept of using Content Based Image Retrieval (CBIR) technique for the purpose of tracking small industrial assets and identifying their location in SKUs. Since small components may be laid in different orientations, there is a need for identifying their shapes from images captured and search the *component image database*. In this paper, we extend our CBIR approach by introducing relevance feedback and knowledge repository to improve the identification of the components, besides using rotation invariant shape feature descriptor. We also present mobile application design methodology for implementing such machine vision based mobile application which can be easily deployed in store keeper’s handheld device for asset location identification purpose.

II. CBIR BASED SOLUTION APPROACH

In this section, we present CBIR based solution to address excess inventory management challenges by providing a mobile application. With the aid of a handheld device, store keepers take the picture of left out spare part on the floor. This picture will be sent to a remote processing system where system will compare the visual features of picture with the pre-loaded image features for all spare parts available in the store room. Proposed system will have a pre-indexed database of all component/parts with their location along with associated component details, images and derived image features. Based on image similarity by comparing image features, system will give the closest matched component images and their location details like store keeping unit ID (SKU ID), rack details etc. Method used for retrieving the location details for a given component picture is further explained in subsequent sections.

CBIR techniques are widely accepted technology for retrieval challenges by extracting the image features and comparing the similarity of query image features against features of images stored in a database. Fig. 1 shows CBIR workflow containing feature extraction steps followed by comparison and ranking. Typically the three major primitive features in any image are color, texture and shape. In this work, we use *shape feature* for image comparison purpose. In our earlier work [6], we introduced a **Fourier Edge Orientation Autocorrelogram (FEAOC)** which is a shape based feature developed for retrieval of healthcare literature documents. In the present context, we need to retrieve the industrial asset images which have clear and distinct edges give explicit shape details; hence we rely on our shape based feature for asset image retrieval. Steps involved in our approach are discussed below.

A. Pre-processing

In the pre-processing step, all the images are first resized to standard size using bilinear interpolation to address variation in resolution. Possible distortions in the images are removed by applying noise elimination techniques. Image enhancement techniques are applied to improve the image quality for improving the retrieval performance. Since the feature of interest in this retrieval exercise is shape, colour information is discarded and all the images are converted to grey-scale for further processing.

B. Feature Extraction

FEOAC is used to describe the shape of industrial components. It is an effective shape descriptor for retrieving binary images, which involves computation of gradient magnitude and orientation using Sobel operator. Resulting magnitude component is thresholded to generate an edge map. Edge Orientation Autocorrelogram [7] is a matrix of 36 rows and 4 columns. 36 bins for 5 degree each for edge orientation and 4 columns corresponds to pixel distance of 1, 3, 5 and 7 apart. Entries of EOAC matrix are populated by comparing each pixel on the edge map with its neighbor to determine their similarity. Two candidate pixels are said to be similar if the absolute values of their magnitude and orientation differences are less than certain threshold value, say *T*. Each edge element is compared with its neighbor 1, 3, 5, and 7 pixels away, and if they are found to be similar, they are added to the auto-correlogram matrix. EOAC matrix is normalized by dividing each element by total number of elements in the matrix. Normalization steps make the translation invariant EOC feature matrix as scale invariant. Finally Fourier transform is applied to feature matrix which makes the feature rotation invariant and helps in reducing the feature dimension.

C. Similarity Computation

Similarity computation is an important step in final stage of CBIR which helps in image ranking. Here query image features are compared with the feature vectors of all images in database with the help of distance/similarity measures. After distance comparison, similarity score is assigned to the database images and then indexed with respect to similarity scores assigned in similarity computation stage.

III. ARCHITECTURE OVERVIEW

In this section, we present a *three-tier architecture* for identifying storage location of small industrial assets. This architecture depicts the process of identifying the rack location of an asset using CBIR. The architecture contains (a) *presentation layer* responsible for visualization aspects such as visualization of asset image, enhanced image, retrieved similar asset images and asset location details; (b) *Business layer* responsible for performing core image processing functionalities and associated computational activities; and (c) *Data access layer* responsible for handling all data management activities related image retrieval process. This includes storage of all asset images available in inventory, feature vectors pertaining to asset images captured during offline process and finally indexing of database based on similarity score. Fig. 2 depicts the high **level** architecture and detailed descriptions of its components are listed in Table I.

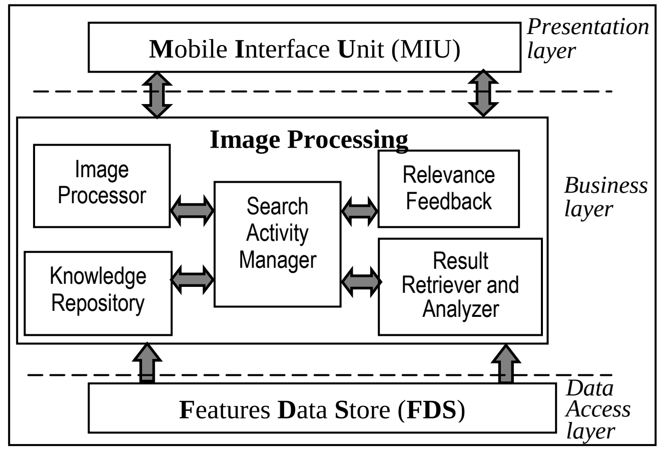

Figure 2. High Level Architectural Design Diagram for Identifying Storage Location of Small Industrial Assets

A. Mobile Interface Unit

This component facilitates capturing query image pertaining to an asset on the assembly floor whose location to be detected and visualizing the retrieved images from database along with the SKU and Rack location details. The configuration manager (see Fig. 3) helps in configuring the screen features and produces the information appropriate to the user.

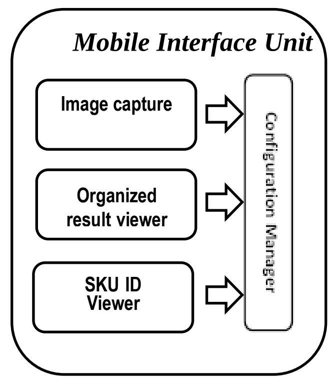

Figure 3. Mobile interface

B. Image Processing Engine

Image Processing Engine performs the business level operations. It consists of five components:

1. *Search Activity Manager*

The *Search Activity Manger* controls the order of execution of the system. It interacts with other four components to co-ordinate the image retrieval activities. *Knowledge Repository* archives recent user activities for analysis purpose and *Image processor* performs image processing activities on both query asset image and database images. *Result retriever and analyzer* retrieves visually similar images based on analyzing the distances between query image and database images.

2. *Image Processor*

Image processor is heart of image process engine unit which receives images as input and performs three steps (fig. 4): (i) *Image Validator* which performs validations of

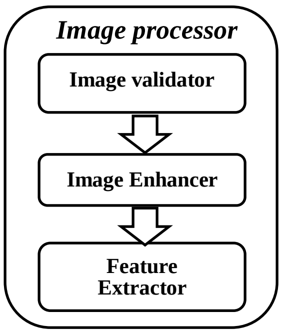

Figure 4. Image Processor unit

Images according to the rules defined in the system and checks the validation rules on all the attributes of the query image such as format, size, resolution and other quality aspects. Custom validations can also be defined in configuration modules as per the user requirement. (ii) *Image Enhancer* performs the basic image preprocessing functionalities to enhance the image prior to feature extraction. The Imaging functionalities such as noise removal using filters and contrast enhancements improve the quality of the images. (iii) *Feature Extractor* extracts the image features using techniques discussed in section II. Image processor modules are useful in two scenarios: (i) for query asset image during online search process and (ii) during offline indexing process where all asset images in the database are processed.

3. Knowledge Repository

Knowledge Repository helps in optimizing the performance of overall system. A small memory is allocated in the system, which contains the data about the recent asset image searches (Fig. 5). Whenever a new query is given to the system, initially it checks for the availability of the result in the knowledge Repository. If it finds in the repository, it transfers the results to the Mobile Interface; otherwise it performs a fresh search to find the new relevant results. This Repository contains the recent result set of *Asset Images* and their *indexes*, which are maintained in a *mapping table*. This technique highly increases the response time and performance of the system when the user makes repetitive searches of component of same kind.

4. Result Retriever and Analyzer

The *Result Retriever and Analyzer* retrieves visually similar images by comparing the distances from query asset image features to database asset image features (see Fig. 6). The *SKU IDs* Database stores SKU location and Rack location details which are mapped to the each image in the database. It selects the top *N* retrieved images and corresponding images location details are rendered on SKU ID viewer in Mobile Interface Unit.

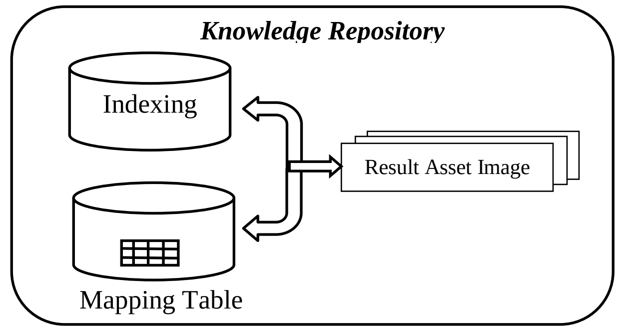

Figure 5. Knowledge Repository unit

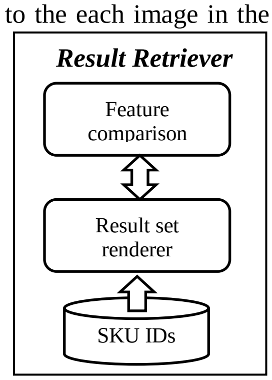

Figure 6. Result retriever & Analyzer unit

5. Relevance Feedback

Relevance feedback is one of the interactive learning techniques. It keeps track of historic search activities and captures the details such as best retrieved results according to the CBIR system and also best results chosen by the user (see Fig. 7). This gives fair understanding of which results are expected by user as best result against the best results given by the CBIR system. This repetitive process is a continuous learning mechanism the system automatically improves the retrieval accuracy using prior knowledge of users.

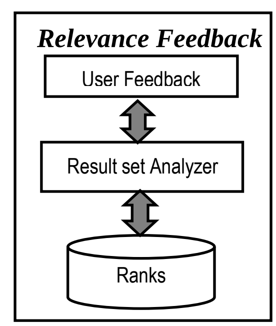

Figure 7. Relevance feedback unit

TABLE I. ARCHITECTURAL COMPONENTS AND THEIR DESCRIPTION

Unit	Components	Description
Mobile Interface	Image Capture	Capture the Asset Image on the Assembly Floor
	Organized result viewer	Display the visually similar images to the end user organized as per similarity score.
	SKU IDs Viewer	Display the rack IDs corresponding to retrieved visually similar images.
Image Processing Engine	Knowledge Repository	Recent user transactions archived for future analysis purpose.
	Relevance Feedback	Store recently retrieved image indices and user actually selected among the top retrieved images as a feedback mechanism to improve the retrieval performance.
	Result Retriever and Analyzer	Retrieves visually similar images by comparing the distances from query asset image features to database asset image features.
	Search Activity Manager	Manages various search activity process as per defined workflow
	Image Processor	Performs core image processing functionalities
Features Data Store	Feature database	Stores the feature vectors of all asset images in inventory

C. *Features Data Store (FDS)*

Features Data Store maintains a database which has feature vectors corresponding to all the asset images at inventory. Typically during offline indexing process, all images are processed using *Image Processor* module to improve the image quality and then extract image features according the approaches defined in the configuration module. These image features are vectors which are stored in the database against each image file name. The *Features Data Store* also stores the asset location details against each asset image file name. Hence there is a one-to-one mapping available for each asset image and their actual location at inventory along with feature vectors of respective images. When a query asset image is captured, it is processed using same image processor and features are extracted and compared with the features in feature database (Fig. 8). Top matched images are retrieved and their respective location details are extracted from the *Features Data Store*.

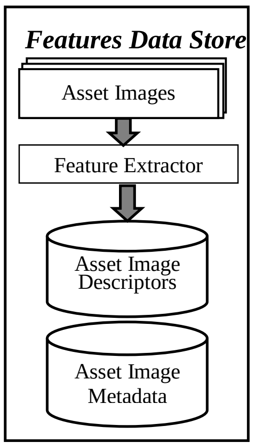

Figure 8. Feature Data Store unit

IV. IMPLEMENTATION

Proposed system is implemented and tested in Android [8] and Android OpenCV [9]. Android is an open source application development platform that comprises of an *Operating System*, *Middle ware* and *key Application*. Android provides a catalyst platform for business to launch useful and innovative mobile applications and helps in creating artistic and enticing mobile applications. It was introduced by Google along with the Open Handset Alliance and has a wide spectrum for social media and other lifestyle applications. This application is built on the system with the following software spectfications - Android Version: 4.0, Operting System: Windows XP, IDE: Eclipse-Indigo, RAM: 1 GB.

A. *Features Data Store*

This phase extracts the features of the entire asset images in the database and stores feature descriptors it in the *Features Data Store*. This process executes when the application is used first time, and also whenever a new set of assets are introduced at inventory. The later is needed to update feature data store with feature descriptors for newly added asset. Along with the features, it also captures the metadata corresponding to the assets. In Android implementation, *SD card* can work as database. This program calls feature extraction algorithm and stores all the feature descriptors in the SD card. Whenever a new image is added in the database, it automatically triggers the features extraction mechanism and updates the database with the features of new images. The pseudo code of the trigger mechanism is as below:

```
On Add<Asset Image> to Database
    Begin
        Activate <Feature Extractor (Asset image)>
        Update <AssetImageVectorsDatabase (Feature Vectors)>
        Update <AssetImageDescriptorDatabase (Descriptor)>
        Update <AssetImageMetadataDatabase (Metadata)>
    End
```

B. *Mobile Interface*

Android provides a rich interface and all the menus can be created through Drag and Drop mechanism. The sytem contains three interfaces: (i) *UI* to capture an asset on the floor, (ii) *UI* to initiate search process and (iii) *Result interface* to display the results. Result interface displays the top N relevant images of the query image along with associated details of the asset.

C. *Search Activity Manager:*

Search Activity Manager controls all the activities in the Business layer. The pseudo code of *Search Activity Manager* is as follows:

```
On call<SearchActivityManager>
        Begin
            Activate <Image Processor>
            Check <knowledge Repository (Image Descriptor)>
            If <found> then
                    Fetch result Images
            Else
                    Assign <Result Retriever component>
                    Fetch result Images
            Invoke <Relevance Feedback Component>
            Store <RankedResultImages> knowledge repository
            Return result images
        End;
```

D. *Image Processor:*

The Image Processor enhances images and computes feature deacriptors of query image and is implemented by using our edge based algorithm – FEOAC. The algorithms involves edge detection and computation of the feature descriptors on the edge map. The psuedo code for this Feature Extraction is as follows:

```
On Event<Feature Extractor (Query Image)>
        Begin
            Normalize Gamma/Color
            Compute Gradients
            Weighted vote into spatial & orientation cells
            Contrast Normalize over overlapping spatial blocks
            Collect FEOAC Over Detection Window
            Return Feature Vectors
        End
```

E. *Result Retriever and Analyzer:*

Result Retriever and Analyzer compares the features of the query image to the features of the asset images in database. Euclidean distance is used for comparing the feature distances in our experiments. A table is maintained with all the euclidean distances of the query image to the database asset images and top N databse images with the least distance are selected as the resultant images. SKU ID database consists of SKU and Rack IDs mapped to the indices of the database asset images. The top N (5 in our experiments) images are rendered by adding its corresponding SKU IDs and returns to the user. The pseudo code for the Result Retriever and Analyzer is as follows:

```
On Call < Result Retriever and Analyser >
        Begin
            Compute Euclidean Distance d (p, q) <query image,
            database asset image>
            Arrange it in ascending order
            Consider first 5 images
            Check <SKU IDs database>
            Retrieve corresponding SKU IDs
            Return resultant images, SKU and rack IDs
        End
```

V. Retrieval Performance

The proposed system was validated on a set of industrial tool images collected from a local vendor. Each image has a resolution of 2 megapixels and was captured from a mobile device. Captured color images were converted to gray scale by averaging the red, green and blue channel intensities. The tool image dataset has 13 classes with a total of 151 images. Labeling is done with the help of a local vendor purely based on functionality of components to ensure that there was no bias due to their shape (feature used for retrieval). Each class has images ranging from 9 to 14 in number (see Table II).

TABLE II. Dataset

Asset Type	No. of Images	Asset Type	No. of Images
Pipe Wrench	9	L-Shaped Angle	11
Spanner	9	Cutting Plier	9
Screw	13	Hammer	15
Cabinet Handle	15	Allen Wrench	14
Door Handle	13	Ring Coupling	13
Tape	9	Screw Driver	12
Threaded Screw	9		

Feature extraction and retrieval is performed as described in section II. Precision and Recall is computed to measure the retrieval performance for the tool dataset. An average precision of 0.8 at recall value 0.9 is observed as shown in Fig. 10. As described earlier, the mobile system is implemented on Android platform. A screenshot of the developed mobile application is depicted in Fig. 9.

VI. Concluding Remarks

In this paper, a mobile system for tracking small industrial asset is proposed. We also presented a detailed software application design for assisting the store keeper in smart asset management. A CBIR framework with our FEOAC feature descriptor is used for finding the location of assets. An image of the asset is captured using the mobile system which identifies and retrieves visually similar images and displays to the user. User can either select one of the images from the retrieved list or select the nearest image by default. Therefore the mobile application can be used both in an automated manner or assist the user. We

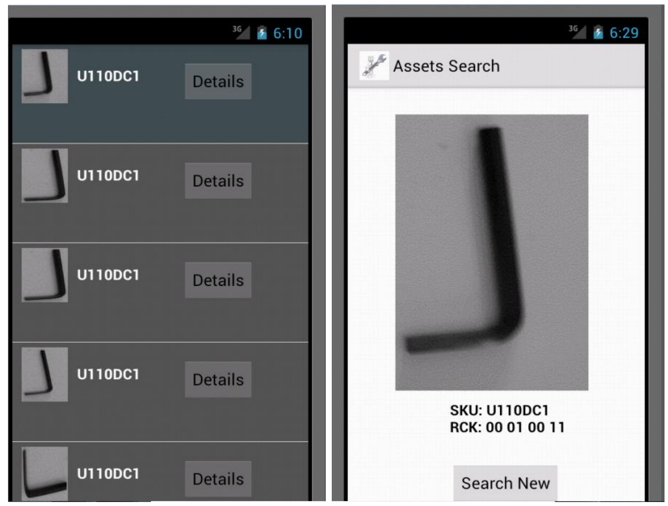

Figure 9. Mobile Application Screenshot

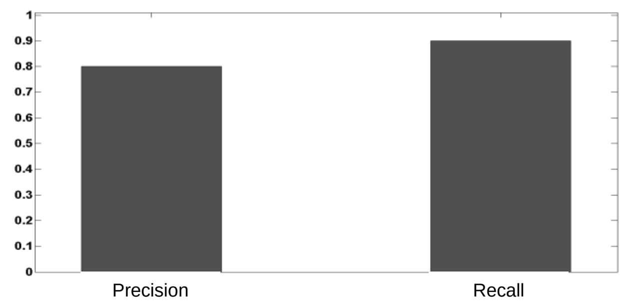

Figure 10. Precision-Recall plot

also described the implementation details of the proposed system on Android platform. Validation of the proposed method is performed on a dataset of 151 tool images and the retrieval performance validates the usability of our system.

References

[1] E. N. Malamas, E.G. M. Petrakis, M. Zervakis, L. Petit, J. Didier Legat, “A survey on industrial vision systems, applications and tools,” Image and Vision Computing, vol. 21 (2), pp. 171-188, 2003.

[2] M. A. Rahman, M. M. Khadem, M. D. Sarder, “Application of RFID in Supply Chain System,” Proc. of the 2010 International Conference on Industrial Engineering and Operations Management, 2010.

[3] G.D. Bhanage, Yu Zhang, Yanyong Zhang, W. Trappe and R. E. Howard, “RollCall : The Design For A Low-Cost And Power Efficient Active RFID Asset Tracking System”, EUROCON The Int. Conf. on Computer as a Tool, pp. 2521-2528, 2007.

[4] N. Talukder, S. I. Ahamed and R. M. Abid, “Smart Tracker: Light Weight Infrastructure-less Assets Tracking solution for Ubiquitous Computing Environment”, 4th Annual Int. Conf. on Mobile and Ubiquitous Systems: Networking & Services, pp. 1-8, 2007.

[5] H. G. N. Rai, K. S. Deepak, S. Syed, and P. Radha Krishna, "A Smart Mobile Application for Identifying Storage Location of Small Industrial Assets," 13th IEEE International Conference on Mobile Data Management (IEEE MDM 2012), July 2012.

[6] W. Chowattanakul, H. G. N. Rai, and P. Radha Krishna, "An Efficient Shape Based Feature for Retrieval of Healthcare Literatures using CBIR Technique," Proc. of ACM Compute 2011.

[7] F. Mahmoudi, J. Shanbehzadeh, A. M. Eftekhari and H. Soltanian-Zadeh, “Image retrieval based on shape similarity by edge orientation autocorrelogram,” Pattern Recognition, pp. 1725-1736, 2003

[8] Android 4.0, http://www.android.com/, 2011.

[9] Android OpenCV, http://opencv.org/platforms/android.html, 2012.